泽众软件测试网
电子期刊

SPASVO

第十四期
2013年2月

心态——新晋测试主管的第一关

代码审查——为可读性努力的巨大能量

什么是合理的开发测试比

怎样搭建与培养自动化测试团队

喂！程序猿，您喜欢测试猿吗？

测试之美---软件测试员的心思你不懂

软件测试团队管理的特点

性能测试：软件测试的重中之重

上海泽众软件电子期刊

2013 年 2 月 第十四期

主办单位：上海泽众软件科技有限公司

联系电话：021-61079698 传真：021-61079698 转 8017

意见反馈：fangmh@spasvo.com 投稿：wangmf@spasvo.com

公司地址：上海市普陀区曹杨路 450 号绿地和创大厦 18 楼 1801 室

邮政编码：200063

公司主页：www.spasvo.com 论坛：bbs.spasvo.com

目录

心态——新晋测试主管的第一关..... 4

代码审查——为可读性努力的巨大能量..... 6

什么是合理的开发测试比..... 8

怎样搭建与培养自动化测试团队..... 10

喂！程序猿，您喜欢测试猿吗？..... 12

测试之美---软件测试员的心思你不懂..... 14

软件测试团队管理的特点..... 18

性能测试：软件测试的重中之重..... 19

泽众软件——专业软件测试门户
2013 年 2 月 第十四期 电子杂志期刊

心态——新晋测试主管的第一关

“如何能有效避免项目延期的发生？”

“研发项目进度延期时，测试应该怎么应对？”

“怎么在需求变化频繁的情况下，保持测试的顺利执行”

最近和很多做测试的兄弟姐妹，关于成长为项目测试负责人时遇到的一些挑战，在网上网下沟通交流。大家纷纷谈到了一些项目过程中的困惑。笔者列举了一些典型的问题。

一开始笔者试图就事论事的讨论每一个待改进点。然而在梳理的过程中，慢慢的发现，这不是一个常见的项目问题，或者管理人员的能力问题——很多的困惑，实际上是一个心态的问题。新的岗位上，岗位本身对业务的要求，对心态的要求，和过往自己岗位的业务、心态偏差比较大。从而对自己产生了迷茫，开始怀疑，产生大量的负面情绪。

借用林正刚林总在《正能量》里面的几句话：“大部分企业调动岗位，通常只注意能力是否胜任，很少注意心态也需要改变。”，“岗位意味着职责，更意味着心态，不同的岗位，对心态的要求是不一样的。”

以测试这个岗位为例。一般而言，项目测试负责人都是从优秀的骨干测试人员中提升的。当这个人的定位是骨干测试人员时，他的测试对象是模块、功能、核心功能级。工作 d 面对相对精准可控的测试内容。

能成为优秀的测试人员，肯定具备如下的素质：思路良好、执行效率高、缺陷遗漏率低，在团队内部有很好的口碑——人不错，有责任心，干活漂亮。这个时候，他的工作环境是正向的，有一种骄傲、责任感的心态，每做一件事情，都会有一种成就导向，加强对自我的认可，甚至还会有一些完美主义的倾向，会越发的让自己的工作成果精益求精。

但是，当他的技能能力满足到上面一个岗位时，开始承担一个完整的项目测试任务，一切都变了。面对的任务不再是相对精准功能模块，而是风险、变化广泛的项目，面对需求变化、周期变化、优先级变化、风险变化等等——研发前端各环节的变化，都会一一体现末端的测试环节。

所以原来美妙的，100%质量的工作成果不太可能再出现，你需要在各种资源和限制的条件下左右平衡，率领团队拿出一个性价比最高的成果，或者相对良好的结果。

原来熟悉的工作模式“给我一个指定的目标，我一定会非常优秀的完成任务!”，变成了“目标天天在变，让我怎么完美的完成目标啊？”的迷茫和困扰。

原来很有成就感，自我认可和团队认可的“这个人干活不错”，变成了负面的情绪、否定和批判“怎么排的测试任务，怎么又漏 bug 了”。

原来的完美主义情结，看到自己亲自做出来的交付件，会感到分外的沮丧和失望。

从就事论事的角度，很很多多的项目管理书籍，甚至网上各种经验心得，都有面对项目的风险如何

控制，如何管理的方法。所以，归根到底，这是一个心态的问题。很多人的能力达到了某个上级岗位的要求，但是心态的高度还没有达到。

同样的问题，在研发的每个环节都会遇到。比如研发模块负责人变成项目负责人，也会经历类似的心态转变，由一个相对积极环境、心态变成一个相对负面的环境、心态。

怎么解决这个问题？

有两个主要的角色，可以从两个途径来解决。

第一、本人。在此阶段，建议阅读很多项目管理，项目风险控制的书籍，理解自己遇到的问题，实际上大部分是业内常见的问题。这样让自己会保持一个相对淡定的心态。而且寻找解决方法的过程，本身就减轻了自己的焦虑感。其次，尝试站在更好的角度去看待质量，看待产品团队对测试环节的要求，承担一些不完美，进行力所能及的优化，这才是从根本上解决此类问题的方法。

第二、更高级的管理者。更高级的管理者自己应该首先明确有一个心态上的转变过程，及时关注，把自己原来的一些经验心得，和新的管理者分享，帮助其快速的调整心态。

心态调整好了，怎么做事情，遇到问题解决问题，相信大家是都知道——毕竟业务能力，职业精神都被认可，才提升为管理者。

代码审查——为可读性努力的巨大能量

代码可读性这个话题一直以来都是备受关注，但是可读性高与不高却没有统一的标准。毕竟各个公司，甚至于各个项目的规范都是不一样的。我们不能说一个抽象性极好，灵活度极高却让人十天半个月都难以搞清楚的代码的可读性高，也不能说一个长达几千行却从头至尾逻辑性比较好的代码的可读性差。那么怎样的代码才算是合理的，才算是可读性高的呢？我想不同之中必有共性，那就是经过审查的、能够被项目组其他成员接受并能尽快看懂的代码就是可读性好的。

为什么要做代码审查呢？

要对代码可读性做审查，这需要人力、物力、以及项目宝贵的时间。对于一个项目来说成本是一个重要的考虑因素，然而审查无疑会增加项目的成本，那么为什么还要做审查呢？其实任何一个项目经理都清楚一个成功的项目都是难以一蹴而就的，开发过程必然会遇到各种各样的问题和阻力，这也验证了那句老话：“软件开发中唯一不会变的就是需求永远会变化”。我们也清楚问题越早的被发现那么损失就会越小，补救花费的时间就会越少，自然成本就越低。但是我们有多大的机会可以尽早的发现问题呢？这不是我们说早发现问题，问题就会跟我们招手说：“看你态度不错，就让你早发现吧！”这么简单的。迭代开发为什么会出现，瀑布式开发为什么难以应对大型的商业、行业项目？思考一下我们不难会发现，客户难以一次性的、整体的、详尽的把自己想要的东西表达清楚，只有当客户看见实实在在的东西之后，他才更明确自己想要什么。好比我们去买裤子，你告诉一个人说：“我要一条简约的牛仔裤”；然后那个人去帮你买，但是具体的颜色你确定么，是黑色还是蓝色？衣服的口袋你确定么，是有扣子的还是没扣子的？只有当你真真切切的到专卖店里面，看到了试过了你才能确定：我要的就是那条 180 的蓝色的口袋上没扣子的 XXX 牌的裤子。也就是说我们很少能够尽早的从客户口中获得问题，除非我们指着我们做出来的东西说：看看，这是不是你想要的。既然如此，要控制的不是尽早的去发现问题，而是如何在问题出现之后尽早的找出问题所在，并解决问题，进而降低项目的成本。

其实软件开发的主要时间是花费在调试上，然而调试中花费的大部分时间又在于读代码。倘若之前开发该模块的人员已远在天边，面对几千行混乱无序的代码任谁都难以承受。因而花费成本在代码审查上是值得的，而且是必须的。可惜的是，现在很少有人去关注代码的规范性、可读性，甚至在大公司都是如此。项目管理者过于注重项目的进度，只要开发者把自己的任务做完了，很少有人去关注他写的代码，甚至开发者自己都不会再去看。

代码审查有何好处呢？

首先代码审查可以提高软件的质量，以及可维护性。这样就可以减少查找错误的时间，提高解决 bug 的效率，提高开发效率的同时降低后期的维护成本。

其次，经过审查的代码是能够迅速被项目组其他成员看懂的，这样有利于项目其他成员更全面的了解业务，对于成员之间交流也有很好的促进作用，当其中负责某个模块的开发人员离职之后其他人员能够迅速的接手相关的开发，并能够尽快的培养新人弥补空缺。

最后，代码审查的过程是总结提高的过程，也是交流的过程，可以有效的提高开发人员的技术水平以及业务素养，增强公司的竞争力，通过总结交流甚至可以从不同项目中提取共性，做出相关产品，从而形成公司自己的核心竞争力，做到行业领先。

如何去做代码审查？

从参加人员来说，应该是项目的整体参与者，如果项目太大，整体参加的成本很高，那么可以以模块为组进行审查。因为他们之间负责的业务是紧密相关的，使用的技术是接近程度比较大的，因而开发的规范应该是统一的。

从审查内容来说，应该是代码的命名规范，以及组织结构。每个项目都有自己的规范，但是如果项目内部使用不同的规范必然会增加发现问题、解决问题的难度同时增加后期的维护成本。

从审查时间来说，应该在每个模块开发完成之后进行，便于开发人员之间交流问题以及体会，并且每个人的讲解时间不要超过 30 分钟，因为模块的业务复杂度不会那么复杂，30 分钟都讲不清的业务逻辑如何保证代码是清晰的。

从审查的结果来说，经过审查的代码应该是参加成员大部分能认同的，并且参加者每个人都能读懂的逻辑清晰的代码，并且通过交流提高项目成员的凝聚力，提高其业务认知度，最好能形成项目之间可以共同使用的产品。

什么是合理的开发测试比

1、首先决定开发测试比的第一因素是公司在当前阶段对该产品所能承受的质量风险

不同公司，对质量风险的承受度是不一样的，同样是做职能手机，apple 和那些"手机中的战斗机"公司显然对质量风险的承受度是不一样的，这不需要解释。其次即使是同一家公司，对不同产品的要求也可能不同，这很可能受到产品的用户群，产品对公司的重要性等因素影响。当然同一家公司在不同阶段对同一个产品，质量要求也可能是不同的，初期阶段，为了抢占市场，抢占先机，质量要求不要，随后产业做大了，质量要求会逐步上升。

2、决定开发测试比的第二因素取决于开发工程师的重要素质：测试意识

不要以为提高开发测试比应该首先提升测试工程师的效率和能力，恰恰相反，首先应该提升的是开发工程师的测试意识和能力。我接触过的众多开发工程师当中，这种规律无一例外：能力越强的开发工程师，测试意识越强。

那些把第一次问题发现由开发工程来负责，测试工程师负责系统性回归测试的团队，总体效率是最高的。

3、决定开发测试比的第三因素是发布流程

经典的风险控制案例是迪斯尼公司发布的一款 DVD 动画片无法播放而被迫召回给公司造成巨大损失。这个发布流程决定了具有重大风险，并且问题遗漏的代价很高。如果我们能够控制分步发布的话，先让一小部分地区试用，然后再逐步扩大到其他区域，即使出现问题代价不会那么高。分步发布其实是线上测试的一种，让真实用户帮我们测试，得到反馈和确认后再进一步发布，如此反复迭代。在迭代中不断增强信心。

其实当今互联网很多产品都已经做到了分布式，那么如果在 release 的时候也能够考虑到分布式发布的话，其实风险可以极大地得到控制。把测试放到线上去，事实证明，效果非常好。有人也管这种过程叫做灰度发布。

4、决定开发测试比的最后一个因素是测试效率

提升自动化率，引入持续集成，等等。可做的事情很多，听起来也很炫，但是。。。我看到过的项目，能够单纯依靠提升测试效率而达到减少开发测试比的，寥寥无几。这种效率的提升，我更倾向于，可以提升工程师的成就感，减轻工作压力。

最后，如何合理估计项目的开发测试比呢？我认为：

1）首先看项目的性质，那些单点的，遇到问题影响范围是 100%的核心业务，应该给予最低的开发测试比，甚至可以是 1:1

2）那些可以有主备模式的甚至是集群模式的，遇到问题影响范围可控的，上线步骤是递进的，比例可以高一点

3）那些测试工程师只负责回归测试和性能测试的，比例可以非常高。但能做到的企业很少。我依然认为这是开发测试最好的合作模式。

请不要迷信 5:1 还是 10:1，也不要鄙视 2:1，3:1。更不要瞎扯我们不需要 QA。引用总书记的一句话：空谈误国，实干兴邦！

怎样搭建与培养自动化测试团队

引：毫无疑问，从企业的立场来看，它期望自动化测试能为企业带来生产效率的提升和测试成本的缩减，说通俗点，就是能用尽可能少的人干尽可能多的事。因此对于那些能够在自动化测试领域做出突出成绩的测试人员，企业从来都是一贯地不遗余力地进行奖赏和激励。因此，在自动化测试领域里，一方面如我们前章所说布满了风险和陷阱，同时另一方面，我们更应该看到充满了很多的机会，对测试人员的职业生涯发展有着至关重要的影响。

好，聪明的你上场了，你正在接管一个正在做手工测试的团队，或者你目前就处于这样的一个团队里，而老板对自动化测试概念又知之不多，不能给予你完全信任的强有力支持，你如何在重重困难中，推行自动化测试实施，而最终取得团队和个人的最大成功？这是我们本章要讨论的重点。

一个好的目标，首先它能够赢得老板的眼球，并有可能逐步转化为老板对你自动化测试实施的支持。

自动化测试项目的实施离不开上级领导的支持，这是一个组织上很关键的因素。因为自动化测试前期的准备要投入人，时间，金钱等资源，比如自动化测试需要买工具，工具则需要培训，而开发工具脚本又需要投入人和时间，如果领导不能在这些方面给予支持，测试人员就真的就成了“巧妇难为无米之炊”，自动化测试的成功更无从谈起了。

所以，在自动化测试的启动阶段，一定要先有一个好的而且可行的自动化测试的目标或想法，它会吸引老板的注意力，并可能获得支持。尤其在自动化测试实施已经比较成熟的企业里，在众多自动化测试解决方案里，一个让人耳目一新甚至拍案叫绝的方案会给老板留下深刻的印象。

但是对于自动化测试刚起步的企业来说，有一些需要特别注意和警惕的地方。这是因为，在知识和经验都不丰富到足以洞察自动化测试本质和规律的时候，很多老板表面上对自动化测试是热情的支持，但实际真实的态度却是底气不足，半信半疑。

我曾遇到过两个极端的例子，一个是某通讯企业的研发总监，在软件开发和测试领域都有深厚的经验，但对自动化测试却有着深刻的怀疑，他认为 QTP 等测试工具并不能真正地从根本上解决测试效率的问题，因此他一直下意识地回避和推迟团队中自动化测试的实施；而另外一个例子是某大型外企的测试经理则是一个技术专家，他对软件自动化测试十分地钟情，几近狂热，认为任何工作都可以交付给程序来做，因此他把自动化测试推到了极致，他的团队开发了大量的脚本和程序，有的只为 demo，有的只为验证 bug。

这两个极端的例子其实是当前软件业界自动化测试实施的缩影，实际上，这两个人的表现更像是同一个人的两面性格，自动化测试上马时盲目乐观，失败后“恨屋及乌”。一番折腾下来，他们对自动化测试是敏感和谨慎的，对于你提出的任何自动化测试目标，他们表面上会支持，实际上更多采用的是观望态度。换句话说，在这种情形下，老板对自动化测试项目的支持是犹豫的和脆弱的。因此，老板是否能够保持对你强有力和持续的支持，不光你要有一个好的自动化测试目标，而是更取决于后续的自动化测试实施能带来实实在在的效益。

【案例】:

测试主管小王打算在自己的测试部门实施系统测试自动化，在经过工具评估后（有关评估详见第三章 Evaluation 一节），他和他的团队决定使用 java 开源的 selenium 做为测试工具。这个想法获得了小王

上级张总的认可和支持。

挑战：小王在着手实施的时候，有如下疑惑和困扰：

1）小王和他的团队没有丰富的自动化测试实施经验，因此，虽然经过了前期的测试自动化效益估算，但对于 selenium 的解决方案到底能否在项目中实施成功，要开发投入多少人力，维护量有多大，小王依然心里没有十足的把握。

2）小王的上级张总是一个雷厉风行的人，他对这次自动化测试的实施也抱有很高的希望，小王如何能够说服张总认识到自动化测试实施的风险，并能给予理解和持续的支持，这是一个要考虑的问题。

对策：小王决定采取以下的措施来最大程度地减小风险，并获得张总的理解和支持。

1）对于第一个问题，由于对测试脚本程序的规模和功能都无法准确预测，小王决定采用快速原型法来开发自动化测试程序，首先在部分核心功能模块中做试点，一边实施一边总结经验，然后再将成功经验进一步推广到整个产品模块。

2） 关于和张总的沟通交流问题，小王决定先准备一个自动化测试的演示程序，邀请张总参加演示会。在演示会上，小王准备了三个演示点，一个是有关自动化测试能替我们做那些工作，一个是自动化测试不能替我们做的工作，另外一个是自动化测试运行中的各种风险和干扰因素。

结果：最后实施的结果是：

（a）小张通过快速原型开发方法，以时间为代价换来了自动化测试实施的稳定和高质量，这为自动化测试的成功实施提供了技术保障。

（b）张总对演示会的内容十分感兴趣，并且和小王约定每隔一个月就进行一次演示会，以便了解自动化测试的状态和进展，并及时解决中间出现的问题。这为自动化测试的成功实施提供了组织保障。

喂！程序猿，您喜欢测试猿吗？

我是一名软件测试员，正如您在其它博文中所看到的一样，我已经罗列出了一大堆有关软件测试的东西，当然这些东西也间接证明了我确实是一名软件测试员。更不幸的是，我还是一名喜欢文字，爱写点散文的软件测试员，如果您只看《祁文之恋》这个系列的一些文章，那么也许您还会认为我还是一个不错的“诗人”。

为啥我来玩博客呢？这还得感谢您们，广大的程序员，因为是你们给了我一份还不错的工作，所以我很喜欢跟您们在一起，就像工作的时候一样，一起在办公室看 Bug 的星星点点，一起数发布的日子。我喜欢您们，所以我就来了你们的领地，但问题是您们喜欢我不？

我对着 Word 问，喂！程序猿，你喜欢测试猿不？

Word 然后回答说，喂！测试猿，这个问题很白痴耶！ 我是 Word 叫我怎么回答你，你自己回答好了！

好吧，我承认这个问题真的很儿童，可问题是年终述职总结会议的时候，我的一名员工当着一群测试猿说，我们得不到程序猿尊重，老嫌我们给他们找麻烦！

老婆说，我有点神经质。我想这可能是真的，身为他们的一个小头头，听见员工这么消沉的观点时，我的脑海里马上想到的是马克思主义官方哲学“矛盾是对立与统一的”忽悠过去。

但不得不佩服，测试总监的敏锐感和嗅觉还是很狗蛋的，因为总监这时发话了，“作为测试人员，如果你不仅能找到问题的所在，并且能够直接修复了这个缺陷，那么程序猿们肯定会佩服你、并且尊重你的”。

老大出口，却实不同凡响，一语中地给那些做过开发的指明了光明的前途，而那些更多不知代码为何物的测试纯猿们却流露出了悲伤的心情，“完了，原来我们在测试猿王中的地位这么低啊”。

也不知我发了那根神经，看着那双双盯住我切急盼望去否定老大眼神的美眼们。怒发冲寇，四川话叫脑壳儿冲动，在老大说完话的 1/10 秒时脱口而出“你怎么能这么说呢，这个观点站不住脚的，因为。。。”，停顿了大约 3s，脑海里面在 google 证据，不过 google 服务器那时刚好间隙性短路，一时打不开，正想去 baidu 试试时，kugou 却播放出了另一个女经理优美的声音，“我也不同意你的观点，要这么说岂不是。。。”。

要说打断女人说话，在文学的绅士学里面是很没有礼貌的，但总监就是总监，总监的权威是不可以挑战的，老大提高了音量“程序猿们都是技术宅，所以只有你的技术，能超过他们，他们肯定佩服你们。这就是为什么微软、Google 的 SDET 那么牛 B，从来没听说过有程序猿不尊重他们的故事啊！ 所以我们要搞自动化测试，自动化测试搞起来了，程序猿就不会说我们不懂技术了。自然，我们就得到了尊重”。

女经理的眼神瞟了我一下，我回了她一个眼神“妞，我懂你的意思，但他是总监，我的头啊”。我真的很怕，尤其是怕这种美女发出的加油的眼神，女经理的电又来了“哥哥，你不剥倒他，谁还敢剥啊”，好吧，死也不过碗那么大个巴拉，本以为会说出啥理性的语言，结果我只说了句“我虽现在找不到理由，但是我还是不赞同你的观点”。明显感觉到了老大要提高声音回答的阵势，还好另一个 MM 聪明地及时说到“说偏离题了，继续述职吧”。

“姐，你真懂哥哥的心啊”。

这事虽然很靠，但却提出了几个猿疼的问题，好像还有那么点意思。

问题 1：程序猿为什么不喜欢测试猿？

问题 2：测试猿靠什么取得程序猿的尊重？

问题 3：难道程序猿真的恐怖到只剩技术？

问题 4：测试猿难道真需要帮程序猿修复 Bug 星星？程序猿的 Bug 被测试猿修改成功后，会有啥心理效果？

问题 5：测试猿为什么需要关注？

。。。。。。

我们是生活在蚁巢中的蚂蚁，俗称蚁族，因为我们太眇少，所以我们希望有放大镜好好看看我们的身体。

不对，我们是猿，上面这句话应该改成下面这样也许才比较适合。

我们是生活在黑森林中的猿，俗称猿族，因为我们太黑，所以我们希望美白。

写到这里，我突然觉得也许下面这一个问题才是急迫重要的，有时间、有兴趣、有电脑、有 CSDN 账号的四有青年，不怕麻烦的话，敬请指教！

“程序猿，你喜欢测试猿吗，如果喜欢，喜欢什么样的测试猿？”

特别申明：文中的内容不一定正确，文中表达的观点也没有经过仔细的推敲，文字也很随意，有时还会放荡，故事也不一定真实，即使有那么一点真实，也有可能放大，各位看客体谅一下。

测试之美---软件测试员的心思你不懂

希望阅读本文的朋友是做过测试并有一定经验的，不然，你明白我说的什么意思，但你对本文并不一定深有体会。

测试人员的定位

这其实是个有趣味且值的问题，包括经常跟测试人员打交道的开发人员，甚至测试人员自己都没弄清楚自己职位到底该如何的定位。当别人问人什么是软件测试时？ 噢！等等，我翻翻书，“软件测试是通过一定的测试方法和工具发现软件的中的缺陷从而来提高软件质量。”

噢？测试发现软件中的所有缺陷么？不能！

噢？测试真的可以提高软件质量么？这个还真不敢保证。

询问者轻蔑的的走开了，处于礼貌，他们可能没有笑出声来，但他们的眼神已经告诉了测试人员答案，测试是个可有可无的工作。留下测试员非常的窝火，但貌似真的找不出非常有力的证据，来证明自己的存在“不可或缺”和“不可代替”的价值。

软件测试人员接受专门的培训来发现并报告问题，他们通过发现和报告软件的异常问题和存在的风险，进而帮助公司、开发团队、客户和最终用户。

那么我们可以把测试人员比作警察吗？在软件开发过程中并没铁定的“宪法”，他们并不能依照“法律”是去“逮捕”任何人，尽管软件开发的世界里完全可以制定出一定的法律。在法律的世界里，一方受到惩罚，一定有另一方面受到的伤害。但软件缺陷不是这样，也许这个缺陷会造成巨大的伤害，也许一定伤害也没有。也许我们的“法律”根本无法评估一个的伤害到底有多大。

好吧！既然不能做警察，那来做法管好了，让测试人员来做“质量把关人”。这其实操作起来很困难，也不太公平。所谓“质量把关人”，就是在软件发布前将该软件看做一个商品。由测试人员来权衡风险、必要性、市场需求和成本开销。噢！测试人员的高度不够，评估和承担风险其实是项目管理者或公司管理层的任务。

到后面可能测试人员已经抛弃了测试人员的本质工作（发现并提交问题），而是花费大量的时间在权衡和评估每一个问题。其实，测试人员清楚地知道不客发现和解决多少问题。软件代码里总是还潜伏着一些问题，所以，他们一般不太情愿盖那个质检合格的红印。这就是说等“质量把关人”去确定产品合格，可能要猴年马月了。

测试人员其实更愿意做侦查取证小组或验尸法医。他们只提取证据。接下来的你们看着办吧。

好吧！软件测试人员的工作远不至这个，以下任何要求都可能决定测试人员的使命，你（测试人员）期望的是哪种要求？

- 快速找出重要的软件问题
- 对产品质量提出总体评估

- 确认产品达到某种具体标准
- 帮助客户改进产品质量和可测试性
- 保证测试过程能够达到可分清责任的标准
- 帮助预测和控制支持成本
- 帮助开发人员完成测试工作
- 参与需求并从测试的角度提高软件的可测性
- 为满足特定客户要求，完成所有必要的工作

对于测试人员来说这太啰嗦（复杂）了，他们只是单纯的喜欢找缺陷（bug），并像探秘一样的把缺陷定位出来。这就像好玩的寻宝游戏。没人事先知道答案，这样对测试人员来说才是有趣的挑战。

测试人员有趣的特质

好吧！为了完成这项有趣的挑战，测试人员应该具备什么样的特质呢？

首先要有好奇心，想弄清楚事物是怎么运行的；其次喜欢动手试验，想知道尝试使用功能演示时不同的用户场景和试验会发生什么。

再次，需要一点胆大精神，不害怕会破坏什么东西，不管你有多位高权重，他们也不害怕把发现的事实告诉你，他们更不害怕站出来据理力争，一定要把他们相信可能影响到产品成功的问题解决掉。

善于分析，善于学习，事实上，测试人员一直在学习，他们的工作性质要求如此。技术总是在变化，接到的每个项目或多或少跟上一个项目不一样。有时候有很好的文档，有时候却没有，必须问出正确的问题，研究正确的问题，把谜题的各个碎片联系在一起，然后得出正确的结论。

当然，测试人员也有不好的特质，尤其对于那些经验丰富的人为说，不容易信任人，这是从实践中历练出来的，别人总是告诉他们模块 X 不需要测试，或代码 Y“没动过”，这种信息错的数多到数不清了。所以，就算你告诉测试人员草是绿的他们也要亲自过目才敢相信。当然了，不是所有的测试人员都具备这些特质。好吧！也许你做测试是为了一份稳定的工作来生活。也许你不是“真正的”测试员。

寻找测试的乐趣

只懂执行其他人测试想法的人，不能算是一个真正意义上的测试人员。当一个测试人员运行一大堆已有的测试用例时，容易心生厌烦。可能会快运行这些测试，只是想让他们从眼前消失，这意味者他们可能不会非常关注执行的测试，当然也就不能像认真彻底的执行者一样找出某些问题。

很多测试人员觉得单调乏味而不屑运行回归测试，虽然大部分测试员都理解甚至同意回归测试的必要性。

一个“真正的”测试人员一定会把这些已有测试看作自己的职责范围，重新考虑其中的想法，提出

问题，充实和改变测试，探究原来的分析没有考虑到的地方。如果原来的分析实在很棒，寻阿能他们也找不出来太多可有更新充实的内容，进而增加了无聊指数。

并非发现的所有问题都可以得到解决

虽然，看到这个结果会打击测试人员的积极性，但这是真的。最有经验的的测试人员会同情地拍拍你的肩膀说：地球人都知道事情不仅仅是发现问题那么简单。他们也会充分理解、会力支持你的决定：问题 A、B、C 可以不解决，并不会有人对这样的决定怪罪你。拥有多年工作经验的测试人员会说出大家都愉悦的意见，因为他们从这家公司的项目经验中学乖了，知道这样会给他们（以及他们部门）带来最好的质量结果。但是需要记住，他们之所以肯牺牲问题 A、B、C ，很可能是为了说服你解决更严重的问题 D 和 E 。当然，大多数测试员是希望发现的所有问题都能够得到处理，现实总没希望的那么好。

测试员只喜欢有趣的缺陷

所有的测试人员都会告诉你，缺陷是存在的，然后缺陷就真的存在了。一般来说，让事情变得好玩并非缺陷的数目。比如一个测试人员可以在大的网站应用程序中发现上千个表面错误，就是语句与错别字，给用户看的文本有语法错误，图标上的颜色不对，或都屏幕上有东西位置放得不对。

测试人员非常讨厌这样的错误，特别是发现有很多的时候。因为记录这类错误比发现它们所花费的时间更长。而且他们一般属于低优先级，很容易得到解决。对！测试人员就是变态的喜欢让开发员束手无策的问题，这样似乎更能体验他们的能力与价值。

不要去预测问题优先级

在 IT 领域经验丰富的前辈会告诉你，某个应用程序的最终用户可能会对你觉得微不足道的问题深切关注。这跟人的“烦恼因素”有很大关系，一个错别字或字体用得不恰当可能不会影响用户的使用，项目组的所有人都认为这是个小问题。但是对于每天要看两千遍的用户来说，“烦恼因素”是非常高的。

例一个双击鼠标就可以完成的操作，我设计成了三击，只是多点了一个鼠标而已。这也许有趣，但对于每天操作几百遍的用户来说，他会破口大骂地拿起鼠标甩到地上。这太令人讨厌了。这影响的他们的工作效率，也行效率与绩效、奖金挂钩。

测试人员报告他们发现的一切问题，其中经验丰富的人员会根据其理解来报告严重程度，但一般来说不要试图预测业务优先级。他们理解中的业务优先级通常就像开发团理解的一样，是不太完整的，并不是基于用户的个人体验做出的。经常有用户愿意“将就”使用有严重错误的代码，却在最后一刻强迫要求修改或添加看起来并不重要的东西。

测试人员的工作是寻找、发现、报告，而不是用神一样的能力去评判，测试人员应该随心所欲的提供他们的专业意见，事实上，项目组的所有人都应该随心所欲地提供专业意见。

报告你发现的所有问题

有一个令人遗憾的现实，那就是测试人员不将他们发现的所有错误报告出来。原因可能有很多，但最常见的是他们觉得报告某一种或某一类错误没有意义，因为反正都不会被解决的。这是从实践中“学”来的，你通常会发现有这类态度的测试人员不抱幻想、厌倦、愤世嫉俗、对工作不感兴趣。他们报告缺陷的兴趣和热情已经被工作环境慢慢消磨掉了。另一个原因也许是他们相信从政治上和实际上来说，报告他们发现的一切东西是不“聪明”的，他们应该只报告那些公司在乎的东西。那么，如果公司看不到

整个大局，怎么知道在不在乎呢？每个人都明白很多错误是不能（或者从财务的角度来说不应该）在产品发布前解决的。成功项目管理的“艺术和工艺”的一个要素是对推迟和解决哪些缺陷做出正确的决策。比如，项目组决定解决 1 4 个缺陷，推迟另外 3 2 个。但是测试人员选择不报告 3 2 4 个缺陷，因为开发团队“从不解决”字段错误，这意味着项目经理和上层的管理者正在根据错误、不全面的信息作决策。在这个例子里，用户界面就不能在万众瞩目的黄金时段隆重登场。

另外，就算是在一个并不解决某类错误的公司，报告每个错误也可能会最终改变公司的政策（或称之为“一直实行的陈规”）。如果一个测试人员报告了 4 0 个错误，一个都没解决，应用程序就发布了，然后用户以紧急的优先级报告同样的错误并要求尽快地解决它们，那么开发团队和项目经理以后就会开始注意这类缺陷

测试员一直在想法破坏你的程序

好的测试人员同时是富有创造力和想象力的。测试通常是一个破坏的过程，正因为如此，在正式产品环境下运行测试需要非常谨慎的决策。好的测试人员不必试图证明软件运行正常，他们是来证明软件不能正常运行的。这一态度差异是测试人员能发现如此多缺陷的主要原因，他们就是想发现缺陷。他们分析手上所有的信息，坐下来思考怎么才能破坏应用程序。项目组里没有其他人有这样的使命。开发人员一般甚至没有足够的时间持续写代码，更不要说试图挤出足够的时间来想怎么破坏代码了。最终用户通常只是执行日常工作的操作，如果有东西“坏掉了”，他们可能陷入恐慌和沮丧之中。另一方面，只有测试人员勇敢地参与进来，使出吃奶的劲儿去发现软件中的缺陷，他们却为发现另开发人员痛苦的缺陷而兴奋不已。

这正好应验了妈妈一直告诉我们的话，要是你只盯人身上坏的一面，那你就只能发现坏的东西。测试人员全面地盯着系统中出错的一面找问题，顺便也就检验了运行正常的部分。但他们关注的焦点总是向着错的东西，而不是对的东西

测试角色的本质

很久之前，就有关于测试人员的角色的争论，我们再来总结性的说说测试角色的本质。

一些人认为测试人员的角色是保证质量，如果有人能决定到底“质量”指什么？ 这个似乎很难说清。

另一些人认为他们的角色是通过训练开发人员的寻找缺陷帮助他们编写出更好的代码----在开始编写的时候就不存在错误的编码；

还有一些测试专家集中精研究为何以及如何找到缺陷：在不同的环境下寻找缺陷所涉及的策略、技术和术语。

所有这些说法都很有趣，在某种意义上都是对业界有溢的。但是，从本质上来说，测试的意义就是发现缺陷。测试人员通过项目组和管理层展示缺陷、问题或瑕疵“保证质量”，进而帮助他们做出更好的决策；他们通过向开发人员展示其代码中的错误，使其知错就改引以为戒，进而帮助他们改进工作；他们学习新的策略和技术以便发现更多的（或者更重要的）缺陷，他们把工作归类到新的策略里，如游历式，进而帮助 其他人发现缺陷。如果在测试过程中没有发现（或者只发现了很少的）错误，那么这也是重要的信息。

软件测试团队管理的特点

凡是在软件公司工作过的人，仿佛都知道开发和测试是对立的，而且开发人员与测试人员的沟通很困难？为什么呢？那样我们简单对比分析一下两种工作的特点。

一个简单的现象，不知道大家是否注意过？英语文化为首的洋人和中国文化为首的东方文化在选择一个事情的时候总是对立的，如果是一个硬币总是选择了硬币的不同一面，很奇怪？例如：中国人姓方在前面，鬼子放在后面； 我们靠右侧通行，鬼子喜欢左面；我们避讳先祖的名字，鬼子把先祖的名字当作自己儿子和孙子的名字；我们写地址先写国家，后写细节，鬼子先写住什么街道最后才是国家；我们过去是竖着写字，从右到左； 鬼子也是翻过来。 凡此种种，只要能对立的地方双方就选择了对立。难道是上帝在作怪么？让两种文化选择正好相反？

其实，无论选择了什么，看起来都是穿衣吃饭，活的不错，也没有因为这些过的难过，其实就是习惯变成了文化，文化变成了特点，特点变成了区别。 对比开发人员和技术人员好象也是如此。由于工作的特点，对待同一个事情上，思维的出发点不一定，经常会选择硬币的两面！ 传说的只能背靠背，无法心连心？ 哈哈！

开发工作的特点是对所负责的部分，模块要做到非常了解和熟练，确保自己负责的部分要没有问题，然后才是整合到真个产品中不发生问题， 当然这个是绝大部分开发工程师的工作内容。

测试工程师呢？首先要了解的是产品的宏观特点，然后是先确保产品的基本有效，才开始逐步确认细节无误。

两个工作的特点正好是微观到宏观和宏观到微观的思维模式。这样的思维模式为基础，导致了在问题出现时的解决思路也不同，个人思维的不同，个人需要掌握的技能也不同，要求的方式也不同，这么多不同必然要导致管理者采取不同思路组建团队，所以我个人的经验是：

管理开发人员是以强调技术深度为导向的方式，必须要对技术的深度有很好的把握和跟进，一个明显的倾向就是开发人员是用技术来衡量领导的能力的， 只要技术压得住，别的都好说。 如果技术不对，别的都是胡扯的倾向。

测试人员是一强调技术的广度为导向的方式，需要对产品的历史，现状，应用，市场，特点，使用习惯等等方面都有所了解，要能进行初步的验证。但是在执行具体的验证过程又是比较容易的（相对于开发人员来说），多以具体操作为主，容易上手。所以测试管理的特点是要求主管更加关注的是人员本身的学习能力和动手能力，如何有效的配置资源，达到效益最大化。

综上所诉，个人认为测试管理者更重要关注的是根据公司的现状，如何建立起来快速有效的，具有学习能力的团队。由狮子带领的绵羊一样可以打硬仗！开发人员必须就是狮群了，每个技术环节的疏漏，必然导致问题的出现！

总结：测试人员要更加理性的强化统筹的方法管理团队。

性能测试：软件测试的重中之重

性能测试在软件的质量保证中起着重要的作用，它包括的测试内容丰富多样。中国软件评测中心将性能测试概括为三个方面：应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。

性能测试在软件的质量保证中起着重要的作用，它包括的测试内容丰富多样。中国软件评测中心将性能测试概括为三个方面：应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。通常情况下，三方面有效、合理的结合，可以达到对系统性能全面的分析和瓶颈的预测。

应用在客户端性能的测试

应用在客户端性能测试的目的是考察客户端应用的性能，测试的入口是客户端。它主要包括并发性能测试、疲劳强度测试、大数据量测试和速度测试等，其中并发性能测试是重点。

并发性能测试是重点

并发性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标和资源监控指标来确定系统并发性能的过程。负载测试（Load Testing）是确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统组成部分的相应输出项，例如通过量、响应时间、CPU 负载、内存使用等来决定系统的性能。负载测试是一个分析软件应用程序和支撑架构、模拟真实环境的使用，从而来确定能够接收的性能过程。压力测试（Stress Testing）是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

并发性能测试的目的主要体现在三个方面：以真实的业务为依据，选择有代表性的、关键的业务操作设计测试案例，以评价系统的当前性能；当扩展应用程序的功能或者新的应用程序将要被部署时，负载测试会帮助确定系统是否还能够处理期望的用户负载，以预测系统的未来性能；通过模拟成百上千个用户，重复执行和运行测试，可以确认性能瓶颈并优化和调整应用，目的在于寻找到瓶颈问题。

当一家企业自己组织力量或委托软件公司代为开发一套应用系统的时候，尤其是以后在生产环境中实际使用起来，用户往往会产生疑问，这套系统能不能承受大量的并发用户同时访问？这类问题最常见于采用联机事务处理(OLTP)方式数据库应用、Web 浏览和视频点播等系统。这种问题的解决要借助于科学的软件测试手段和先进的测试工具。

举例说明：电信计费软件

众所周知，每月 20 日左右是市话交费的高峰期，全市几千个收费网点同时启动。收费过程一般分为两步，首先要根据用户提出的电话号码来查询出其当月产生费用，然后收取现金并将此用户修改为已交费状态。一个用户看起来简单的两个步骤，但当成百上千的终端，同时执行这样的操作时，情况就大不一样了，如此众多的交易同时发生，对应用程序本身、操作系统、中心数据库服务器、中间件服务器、网络设备的承受力都是一个严峻的考验。决策者不可能在发生问题后才考虑系统的承受力， 预见软件的并发承受力， 这是在软件测试阶段就应该解决的问题。

目前，大多数公司企业需要支持成百上千名用户，各类应用环境以及由不同供应商提供的元件组装起来的复杂产品，难以预知的用户负载和愈来愈复杂的应用程序，使公司担忧会发生投放性能差、用户遭受反应慢、系统失灵等问题。其结果就是导致公司收益的损失。

如何模拟实际情况呢？找若干台电脑和同样数目的操作人员在同一时刻进行操作，然后拿秒表记录下反应时间？这样的手工作坊式的测试方法不切实际，且无法捕捉程序内部变化情况，这样就需要压力测试工具的辅助。

测试的基本策略是自动负载测试，通过在一台或几台 PC 机上模拟成百或上千的虚拟用户同时执行业务的情景，对应用程序进行测试，同时记录下每一事务处理的时间、中间件服务器峰值数据、数据库状态等。通过可重复的、真实的测试能够彻底地度量应用的可扩展性和性能，确定问题所在以及优化系统性能。预先知道了系统的承受力，就为最终用户规划整个运行环境的配置提供了有力的依据。

并发性能测试前的准备工作

测试环境：配置测试环境是测试实施的一个重要阶段，测试环境的适合与否会严重影响测试结果的真实性和正确性。测试环境包括硬件环境和软件环境，硬件环境指测试必需的服务器、客户端、网络连接设备以及打印机/扫描仪等辅助硬件设备所构成的环境；软件环境指被测软件运行时的操作系统、数据库及其他应用软件构成的环境。

一个充分准备好的测试环境有三个优点：一个稳定、可重复的测试环境，能够保证测试结果的正确；保证达到测试执行的技术需求；保证得到正确的、可重复的以及易理解的测试结果。

测试工具：并发性能测试是在客户端执行的黑盒测试，一般不采用手工方式，而是利用工具采用自动化方式进行。目前，成熟的并发性能测试工具有很多，选择的依据主要是测试需求和性能价格比。著名的并发性能测试工具有 QALoad、LoadRunner、Benchmark Factory 和 Webstress 等。这些测试工具都是自动化负载测试工具，通过可重复的、真实的测试，能够彻底地度量应用的可扩展性和性能，可以在整个开发生命周期、跨越多种平台、自动执行测试任务，可以模拟成百上千的用户并发执行关键业务而完成对应用程序的测试。

测试数据：在初始的测试环境中需要输入一些适当的测试数据，目的是识别数据状态并且验证用于测试的测试案例，在正式的测试开始以前对测试案例进行调试，将正式测试开始时的错误降到最低。在测试进行到关键过程环节时，非常有必要进行数据状态的备份。制造初始数据意味着将合适的数据存储下来，需要的时候恢复它，初始数据提供了一个基线用来评估测试执行的结果。

在测试正式执行时，还需要准备业务测试数据，比如测试并发查询业务，那么要求对应的数据库和表中有相当的数据量以及数据的种类应能覆盖全部业务。

泽众软件工具使用技术支持

电话：021-61079698

Email：sales@spasvo.com

QQ：1404189128

MSN：spasvo_support@hotmail.com

产品租用		
下载	在线申请	详细

AutoRunner 是一款自动化测试工具。AutoRunner 可以用来执行重复的手工测试。主要用于：功能测试、回归测试的自动化。它采用数据驱动和参数化的理念，通过录制用户对被测系统的操作，生成自动化脚本，然后让计算机执行自动化脚本，达到提高测试效率，降低人工测试成本。

在线体验		产品租用	
企业版	免费版	在线申请	详情

TestCenter 是一款功能强大的测试管理工具，它实现了：测试需求管理、测试用例管理、测试业务组件管理、测试计划管理、测试执行、测试结果日志察看、测试结果分析、缺陷管理，并且支持测试需求和测试用例之间的关联关系，可以通过测试需求索引测试用例。

其他测试工具

Precise Project Management

Terminal AutoRunner

PerformanceRunner

有关培训、产品购买及试用授权方法等事宜

电话：021-61079698

Email： sales@spasvo.com

QQ：1404189128

MSN： jennyding0829@hotmail.com

SPASVO